blender art MAGAZINE

blender learning made easy

Blender for Architecture and Games Special!!

- Blender Normal Mapping
- ArchViz Tips & Tricks
- From 2d CAD to Blender
- Making of The Cathedral
- Making of Burly Brawl

Amazing Blender

Sandra Gilbert

Managing Editor

It always amazes me the number of uses that can be found for Blender. From simple modeling to full-length animations, Blender has something for everyone. So it should be no surprise that Blender is also used for Architectural modeling and for the production of games.

Blender is a wonderful way to visualize buildings and plans. It takes a little forethought and planning, but then again so does any project worth undertaking. Not only is Blender being used to model and texture Architectural plans, it is also being used to provide walkthroughs, both interactive and standard tour-type animations. Which, in a way is where the game engine comes in. Blender's game engine has been around for quite some time, being used for a variety of games, web-based content and interactive content. Walkthroughs of various worlds, game environments and lately Architectural projects have become a growing area of interest. What could be more cool than setting up your project and then being able to have prospective clients, friends, and family be able to move around at will to check it out.

That being said, the game engine in and of itself, is still used more often for the production of games than anything else. There is a growing section of our community that actively supports and uses the game engine to express their creativity through games of all types. Over the last year the game engine has seen a lot of upgrades and improvements, making the art of gaming easier and more appealing to those who have not yet attempted its mysteries.

So whether you are into Architectural modeling and visualization or, have an overwhelming desire to produce the next breakout game, this issue is going to introduce you to some of the steps needed to get you started.

Happy Blending

-sandra@blenderart.org

blender art MAGAZINE

EDITOR/DESIGNER
Gaurav Nawani gaurav@blenderart.org

MANAGING EDITOR
Sandra Gilbert sandra@blenderart.org

WEBSITE
Nam Pham - nam@blenderart.org

PROOFER
Kernon Dillon

WRITERS
Yellow
Mike Pan
Samo Korosec
Tiziana Loni
Roland Plüss
EnzoBlue
Zsolt Stephan

COVER ART
Sebestian Koeing

Click on the title or page no, to jump to article.

Architectural Visualizations In Blender

More and more architectural students and professionals are finding their way to Blender and the possibilities that Blender offers for quality still images and fly-by/walkthrough presentations. When it comes to using Blender for architectural mockups, there seems to be an ongoing debate over the best way to use Blender and whether or not it is even possible.

There seems to be two main camps in this debate. As a pure visualization tool, Blender is more than adequate to the task. This side of the debate concentrates on Blender's ability to provide a fairly realistic vision of the architectural project for use in showing a client how the building/house/project will look when finished, complete with textured materials, lighting and possible fly-bys/walkthroughs.

The other side of the debate is concerned with accurately reproducing the project from CAD files to create exact replications of the project. While this side of the debate is still working on the best method of achieving this, there has been a lot of ground covered on workarounds and methods for creating fairly realistic models from a variety of file formats.

Following are some tips to help you get started on your project. Most of the tips are taken directly from existing threads at www.blenderartists.com

- Instead of CAD files, try scanning in the plans. It is fast, although not as accurate as a CAD file, but produces acceptable results for visualization purposes.

- Using the snap command (Shift + S) can be helpful in the beginning stages of creating the model, import your plans, snap cursor to first point, command click to next point, snap cursor, repeat.

- File formats to use: dxf and .wrl (VRML1.0), these file formats have been shown to work, although they often need some cleanup.

Converters:
dxf – CAD ProgLT 2006
CAD – ACME CAD Converter

There can be import issues with BIG dxf files. If possible, break the file into smaller chunks and import them separately.

Here's a way to use measurements from Blender without needing Object snaps.

- Spacebar add mesh plane, select 3 vertices X delete vertices, Select remaining vertice, E extrude (x,y,z/axis), numeric input for distance = = line

- Select vertice, E extrude (x,y,z/axis), numeric input for distance. = = snap from endpoint.

- Select two vertices W-subdivide, "missing distance", E extrude (x,y,z/axis), numeric input for distance = = snap to distance from endpoint.

- Select two vertices Shift-D duplicate (x,y,z/axis), numeric input for distance = = parallels

- Adjusting for the size of a circle seems to take some outside object as a reference, the same with ovals.

For more tips and workarounds, do a search on www.blenderartists.com, use keywords: archiviz, CAD, architectural walkthroughs. This topic has been the subject of a great deal of discussion generating many workable methods of achieving accurate models and realistic visualizations. ■

The Game Engine

Since its arrival in version 2.00, the Game Engine has been a source of both frustration and delight for a dedicated hardcore section of our community. Long placed on the back burner in favor of more pressing coding, the last several versions of Blender have seen a return of interest by coders with new features being added.

One of the latest additions is a new physics engine (Bullet), allowing for more realistic rigid body physics, written and implemented by Erwin Coumans. Erwin originally developed the game engine for Blender 2.00 and, after a long absence, he has come back to implement part of the Bullet Physics Library he is developing for the Sony PlayStation 3 into Blender. Below are some tips, taken from the 2.42 release notes to best use Bullet to its full advantage:

- **Don't scale your objects.**
Apply scale using CTRL-A. If you use Sphere-bounds, make sure the radius fits the sphere:

- **Keep masses for dynamic objects similar.**
If an object of 100 kg rests on an object of 0.1 kg, the simulation will have difficulties. In order to avoid large mass ratios, keep mass of objects similar (in the order of a few kg)

- **Assign the right bounds type.**
For a cylinder, choose cylinder, even for non-moving objects. Similar for a box etc. Convex Hull can approximate meshes for moving objects and static objects.

- **Don't use too many vertices in Convex Hull meshes**
About 4 to 32 vertices should be fine.

- **Leave the center of the object in the middle of the mesh**
See this picture: The center (where the axis are) needs to be inside the mesh, not near to the boundary or outside!.

- **Leave the gravity to 10, don't make it too large**
The physics simulation works better with smaller gravity, so if possible don't use large gravity.

- **Avoid very small dynamic objects (< 0.2 units)**
Don't make dynamic objects smaller then 0.2 units if possible. For the default gravity, 1 unit equals 1 meter, so any 'side' of the objects should be bigger then that.

- **No large objects**
Don't use large objects, or large triangles.

- **Don't use degenerate triangles**
Triangles that have extreme long sides as well as extreme short sides can lead to problems.

- **After a few seconds, the object doesn't move anymore. It doesn't interact with moving platforms etc.**
You can manually activate an object, using python command 'object.restoreDynamics'.
Or use the 'no sleeping' button. But don't use the 'no sleeping' button too much, perhaps just for the main character/car etc.

- **Adding dynamic objects as children doesn't work**
Dynamic objects should not have a parent. If you need to add a batch of rigid bodies, check out the addObject2.blend demo, it uses the 'instantAddObject' on the AddObjectActuator.
If you need a complex setup with constraints, like a ragdoll or vehicle, you cannot create that group using an AddObjectActuator. Either use Python for this, to setup the constraints after adding the objects, or wait for future 'constraint user interface' in Blender. ■

For more resources on learning how to use the Game Engine

Download the 2.42 Physics demos
http://www.continuousphysics.com/ftp/pub/test/index.php?dir=blender/&file=physics_demos-2.42-preview34.zip (3 Mb)

Summer of Documentation: Mal - Introduction to the Game Engine
http://mediawiki.blender.org/index.php/Blender-_Summer_of_Documentation

Gamekit Download: Downloadable documentation from the e-shop, explaining how to use the Game Engine
http://www.blender.org/eshop/product_info.php?products_id=83&PHPSESSID=3ae6193f0a9d0e8d1f64218984ae6a17

Blender News

The Department of Architecture in Innsbruck has shown interest in switching to Blender for their design needs. What makes Blender somewhat useless for Architects right now is the lack of numerical input like they're used to from CAD applications and the lack of proper NURBS support (and some more minor details like file format support). They know that these are not trivial to add, so the following idea is being worked on at the moment:

Bring together a group of students and professors from the Department of Architecture, Department of Mathematics, Department of Geometry and Department of Computer Science, invite a group of (core) Blender developers and organize a project where all the "missing" features are added. This wouldn't be a Sprint but more along the lines of Project: Orange, lasting two or more months and having a CAD-development focus instead of an animation one.

The whole thing could be sponsored through the Tyrolean Research Funds and the relevant department leads have already signaled an interest and one of the assistants working at the Department of Architecture is working on a proposal now. The whole thing would happen around spring 2007 and is somewhat dependent on the success of the results we achieve with the Sprint. I was asked to ask the Bf-committers about whether there is enough interest from the Blender developer community to actually participate in such a long project on-site in Innsbruck.

[Bf-committers] Invitation to the Blender Sprint Innsbruck + possible future project in Innsbruck ■
http://projects.Blender.org/pipermail/bf-committers/2006-August/015599...

• Blender at SIGGRAPH

This summer has seen a lot of activity in the Blender Community. With the wrap-up of Project Orange, work went full speed ahead to finish up the 2.42 release of Blender, in time for SIGGRAPH. Blender was part of the Open Source Pavilion at the Boston Convention & Exhibition Center, where SIGGRAPH took place this year. There was lots of excitement to be had for those lucky enough to attend. For those of us that were unable to go, b@rt was nice enough to provide a blog of the daily happenings and videos of the demos that were provided from the Blender booth. If you have not checked those out yet, they can be found at

http://www.blendernation.com/
Search for SIGGRAPH to find the daily blogs and the video downloads.

• The return of the Google Summer of Code

As the summer is now wrapping up, so are the three projects approved for this summer. And while three isn't a big number, the three that were chosen, look to be feature packed improvements that we will all enjoy. Following are the approved projects and links to further information on their progress:

• Interactive Sculpting with Multi-Resolution Models - by Nicholas Bishop

Mentor - Jean-Luc Peurière
http://sharp3d.sourceforge.net/mediawiki/index.php/Google_SoC

• Modifier Stack Upgrade by Benjamin John Batt

Mentor - Daniel Dunbar
http://mediawiki.Blender.org/index.php/User:Artificer/SummerOfCode2006

• Sky Generator by Dmitriy Mazovka

Mentor - Kent Mein
http://projects.Blender.org/pipermail/bf-committers/2006-April/014374.html

In addition to the three projects approved through the Blender Foundation, there is a fourth accepted through Python. The Python Foundation accepted a proposal submitted by Palle Raabjerg to create exporting/importing tools for Soya3D, a Python game engine project. More information can be found for this project at http://www.soya3d.org/wiki/Soya/BlenderTools

• Blender Summer of Documentation

With Blender being improved and the rate of new features being added, keeping up with current documentation has been difficult to say the least for the documentation team. This summer's projects aim to update our documentation and provide a more in-depth look at some of Blender's newer features.

As a complete Blender information junky, I have been eagerly awaiting the completion of these projects and am happy to say that they are progressing nicely. Following is link to the mediawiki where you can download the current version of each project. ■

http://mediawiki.blender.org/index.php/BSoD

BLENDER NORMAL MAPPING

- by Tiziana Loni (TiZeta)

Roland Pluss (Odjin/DragonLord)

Prerequisites

Ability to model, unwrap/uv-map, setup a color and a bumpmap texture channel in Blender.

What is normalmapping?

In practice, it is a sort of advanced bumpmapping. You need two models, a simple low-poly one and a high-poly, detailed version of it.
With the right tools, you'll be able to make the low-poly model look very similar to detailed one, by simply applying a normalmap texture to it. The normalmap comes from the detailed version of the model.

Blender and DENormGen

DENormGen has been developed by DragonLord, as a tool for "Epsylon - The Guardians of Xendron", the game he's developing:

You can download it from the link below, the current version is 1.4.

http://rptd.dnsalias.net/epsylon/

You will get the standalone program, DENormGen, and a script for Blender.
Once installed, in the "export" menu in Blender, you will find a "Drag[en]gine intermediate model" export option.

Model setup

You need to create your two versions of the model, in the same .blend file. For example, in image 01, the first one is 1,508 polygons, while second one is 139 polygons (triangles).

In both versions, don't forget to check and adjust the normals: select "show normal" from the edit panel [F9], select all [A], then press [Ctrl + N].

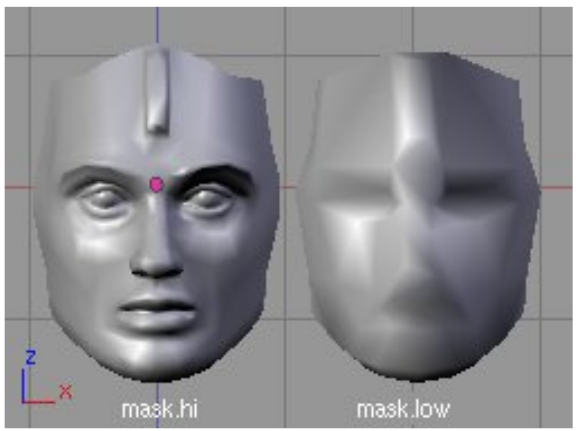


Image1: High-poly and low poly-model

Also, take care to not have unused or doubled vertices around. From DENormGen 1.4 release, the low-poly version name must end with .low (ex. "character.low"), while the high-poly one must end with .hi (ex. "character.hi").

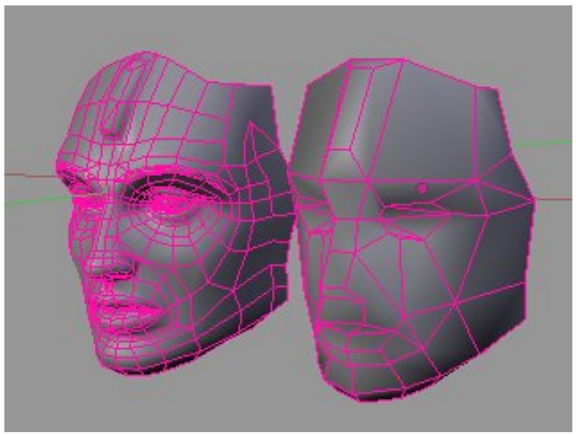


Image2: Wireframe of the model

Another important thing is that the low-poly version must have a material with an image assigned and mapped to uv, a good uv layout and a texture image assigned to it.

You can overlap different UV in a single UV map, to lower the texture size, but you may not have flipped stuff. The image can be whatever you want, at this stage it's only needed to define the image size and run the script.

You can also use a testmap: from the UV editor, enter menu **IMAGE > > NEW**, select the size of the texture and check "UV test grid" button, then push OK. Save the image **IMAGE > > SAVE AS...**

and load it both in the uv layout and in the first channel of the material's texture panel. The image could also be the final color texture, if you already have one.

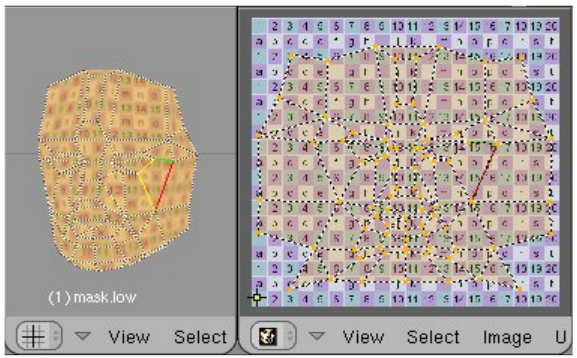


Image3: UV Editor

A better choice would be a grid testmap with numbers and letters on it, to avoid flipped pieces in the uv map. These are the types of images that are used (see images 3-5).

For the detailed version, only a pair of tricks are needed. Together with the .hi suffix, pay attention to proportions. The two versions must be very similar, except for details. General proportions must be the same. This is easy to obtain if you model the low-poly version starting from the high-poly version, or vice versa. It doesn't matter if you make the low or high-poly version first, it just depends on which method fits better for you.

You can also make the high-poly version first and then, use the decimator. But, this will give you a very messy and triangulated mesh. And remember that you don't need to unwrap and add a material to the detailed version. You just need to sculpt it.

Image4: Material Panel

Image5: Texture Panel

That's all at the moment. You have your low-poly and high-poly version in the same file, right?

You've checked the normals and setup the UV and material for the low-poly version.

So, go to Object mode and select both models.Just run the script **FILE > > EXPORT > > DRAG[EN]** gine intermediate model and you will get the .dim file.

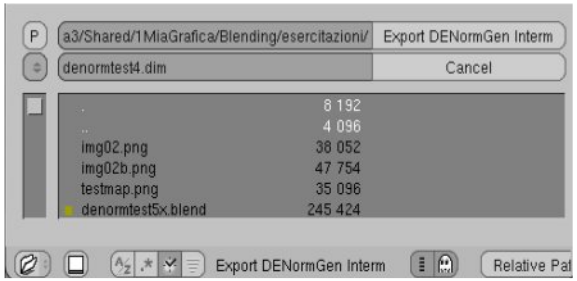


Image6: DENormGen Script running

Now run the DENormGen standalone application and open the file.

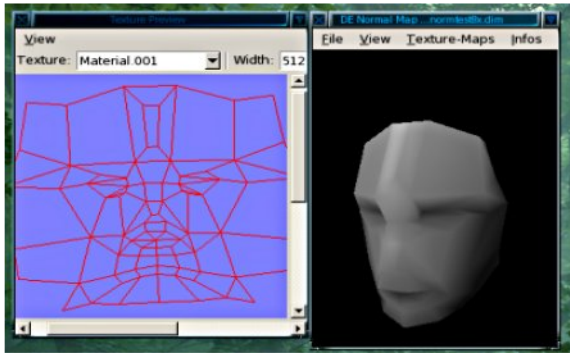


Image7: DENormGen Interface

You can rotate the view with your left mouse button, and zoom with **[SHIFT+RIGHT MOUSE BUTTON]**.

[Alt]+left mouse button moves the light.
From the "View" menu you can switch to the high-poly model and back to the low-poly model.
You can eventually change the map size using the menu: **TEXTURE-MAPS>>TEXTURE MAP SIZES**.

Note that DENormGen also allows you to setup subdivision surfaces for the high-poly model.
Just press [Crtr+T]

Now it's time to make the texture

Enter menu Texture **MAPS>>GENERATE NORMAL/DISPLACEMENT MAPS**.

You have "tangent space" and "object space" radio button options. The first is good for animated objects, while the second is for non-animated objects like walls, buildings or statues.

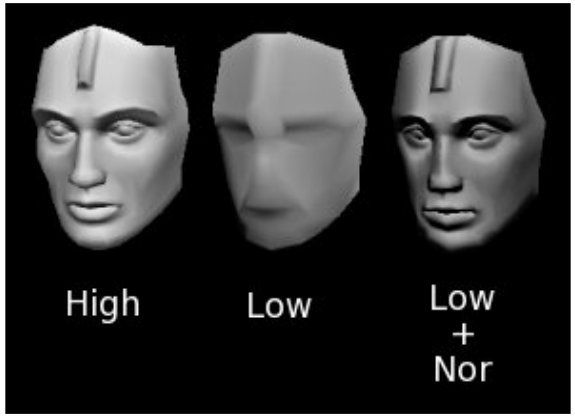


Image8: Detailed model, low-poly and low-poly with normalmap on, as they appear in the DENormGen preview window.

Checking or unchecking "Smooth hi-res mesh normals" allows you to get a smooth or faceted version of the normalmap (in most cases, leaving it checked is ok).

After making your choices, wait for the maps to be generated. Once the maps are generated, use the "View" menu options to see the result of the normalmap applied to your low-poly model, and compare it with the high-poly version.

The normalmap itself is shown in the "texture preview" window.

Note that DENormGen is also able to generate a grayscale displacement map, but this is still in development and may give strange results.

The last step is to save the map: select menu **TEXTURE MAPS>>SAVE NORMAL MAP**. You will get a TGA (Targa image file format) copy of the normalmap. This generated image will need a little tweaking to be used in Blender.

GIMPing the image

The generated normalmap is usable with most common game engines "as it is" but, if used in Blender, it causes light and shadows to be faced the wrong way.

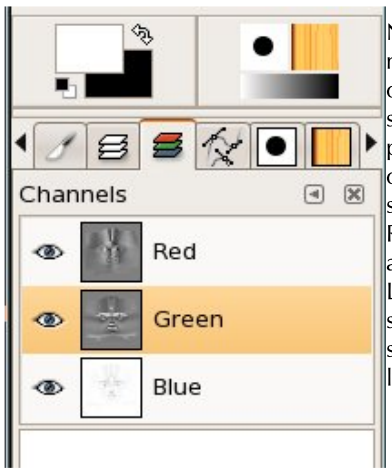
This will be fixed with an option to be added in the next DENormGen releases, but at the moment the GIMP, or similar program, can be used to make it Blender-usable.

If the normalmap is generated in tangent space:

Open the file in the GIMP, and select the green channel from the "Channels" panel.

Pay attention to the other channels layers, they may not be selected or it won't work. They are selected by default, so click on the red and blue channels to deselect them.

Then, invert the colors for the green channel using the Layer menu: **LAYER > > COLORS > > INVERT**.



Note that if the map turns green or something similar, you probably had the other channels selected. Reload the file and try again. Layer must be the same shade, with swapped lightening:

Image9: Select Green Channel only

Image10: What you might get

If you used the object space option:

Like above, but you must invert the red channel instead of the green one.

Setting up the normalmap in Blender

It's very similar to adding a bumpmap, except that you must enable "Normalmap" in the Texture panel too, not only in the Material panel.

Go to the Texture panel **[F6]**, create a second texture channel and set its type to "Image", then assign the normalmap to it.

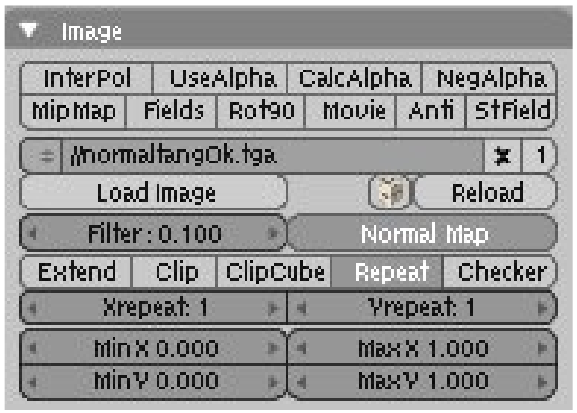


Image11: 'Image' tab in the Texture panel

Important: in this panel you need to press the "Normalmap" button, too. Change to Material panel **[F5]**, and from the "Map Input" tab select UV. Also select Nor from the "Map To" tab.

Be aware that map to "Col" is not selected for this texture channel.

Image12: 'Map to' tab in the Material panel

That's all: fix the first texture channel, in case you've been using a testmap, and give it a good colormap or just a plain tint.

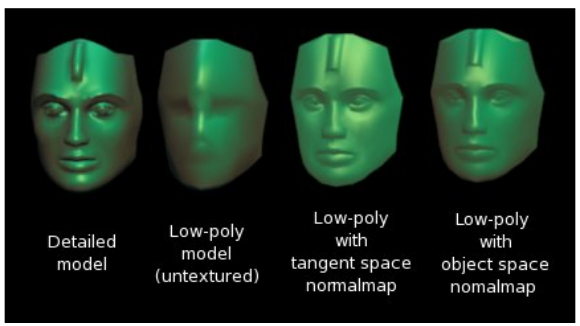


Image13: Rendered Normal Map

Of course, you can also use the remaining channels to add whatever you want, the number of available free channels is the only limit.

Fix the lights, set the camera... and enjoy the render!

What about SharpConstruct?

SharpConstruct is very useful for adding and refining the details of the high-poly version. To use it, export the high-poly version as an OBJ file, open and tweak it in SharpConstruct then, save it again as an OBJ file.

Import the OBJ file back into Blender and Append the low-poly version in the same file.You can also use some pretty good Python scripts to refine the high-poly version, like B-Brush or Expresso.

Troubleshooting.

Even if still in development, DENormGen really works greatly. The only issue I noticed is with low-resolution images, with sharp details: in this case, the pixels can produce some artifacts on the mesh. If you don't want to augment the resolution, you can try to fix it in GIMP:

Select, with the Lasso tool, the jagged edges and apply the Blurring script. For the best result, be sure to activate “Feather Edges” in the Lasso settings, with a fitting pixel radius.

Do not use the Smudge or the Clone tools: what you can do on a normalmap is really restricted, and you could easily ruin the texture. ■

Some Useful Links

Project Epsylon
http://rptd.dnsalias.net/epsylon/

SharpConstruct
http://sharp3d.sourceforge.net/mediawiki/index.php/Main_Page

Espresso and Pytablet:
http://members.fortunecity.de/pytablet/

Fox libraries: (If you use Linux, check your package manager first)
http://www.fox-toolkit.org/

TIPS AND TRICKS FOR ARCHVIZ IN BLENDER

- by Zsolt

Modeling

No N-gons, what now?
One limitation that comes up in rendering architecture is that Blender lacks N-gons. N-gons are polygons with more than three or four sides. While quads are great for many types of modeling (allowing clean subdivisions, loop modeling, etc.), large, complex flat surfaces are easier to handle as one single polygon instead of dozens of small triangles/quadrilaterals. For example, see the wall in the first image. The outline looks harmless enough; it's the facade of a Venetian building, with a place for one door and several windows. This could be solved with one concave (i.e. has holes in it), or several convex N-gons. But, in Blender, we need the following very ugly setup of 113 faces (Figure2). Not only ugly, but it is now hard to continue any sort of modeling with it, like subdivision, adding more detail, beveling some corners. Solution?

1. F-gons

F-gons are fake N-gons. Basically, you select the faces you want, hit F and select Make F-Gon. From now, Blender will represent it as one face when in Face Select mode. It is now easier to select the whole wall. However, this is just a cosmetic change, the above mentioned modeling problems continue to exist. Therefore, the one really good use for F-gons is at the very end of the modeling process.

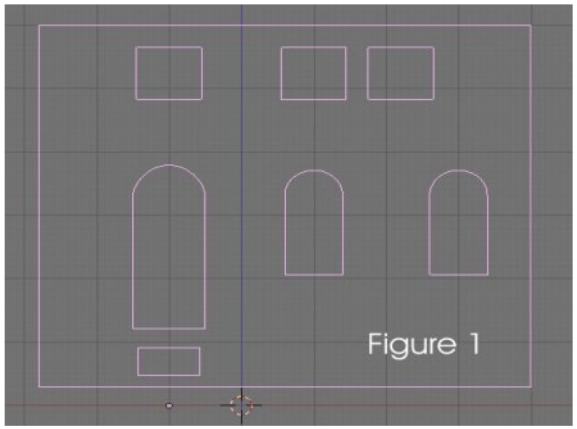


Image1: The wall plan

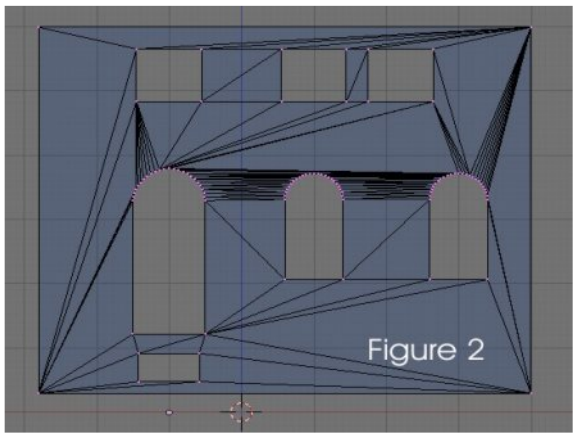


Image2: Wall plan filled with Fgons

2. Go modular!

If you're like me, and prefer "vertex-pushing" as the only true modeling technique, you can try a modular modeling approach. Basically, divide the building into sections, both in height and width, along strategic lines. For instance, if the windows are spaced one every two meters, then one section would be two meters wide, and would contain one window. The height of the sections can be one floor (or smaller if there is more detail). See Figure 3. Notice that the model is not only cleaner but, each and every window can be separated and taken out, and another one put into place, without disturbing the rest of the mesh.

Quick tip: For modeling one section: Add a plane, and move the corners to the proper positions (always, ALWAYS use snap to grid, or type exact numbers by hand). In the same mesh, add the outlines of the holes, inside the plane. Select all, and press Shift-F (Fill). This creates the necessary triangles. Now press Alt-F (Beauty Fill), which will rearrange the triangles to minimize extremely long polygons. Note: you might need to press Alt-F several times for the optimum result. As a final step, convert as many triangles to quads as possible with Select Faces = > Alt-J. Repeat for all sections.

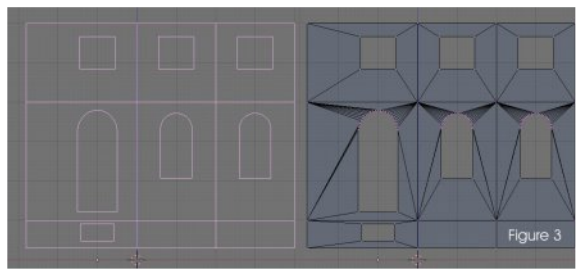


Image3: Beauty fill

This type of modularity is especially useful for large buildings, you can pre-model several window, door types, and then simply duplicate them (Shift-D) and move them around. The exact sizes and the Snap to Grid option ensure they will match up at the edges. When done with the mesh, remove the duplicates (in the W-key menu). And remember to extrude the wall to the proper width!

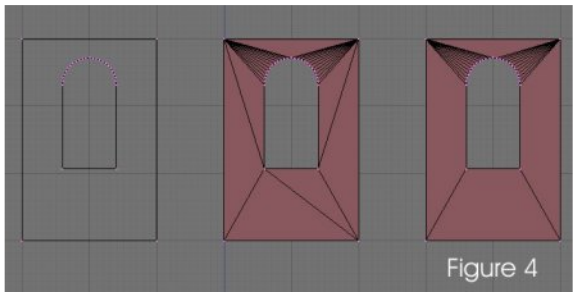


Image4: Small mesh parts

Many other types of models can benefit from modular modeling, which is basically: interchangeable mesh parts, all with given outside dimensions so they can be easily joined or separated.

3. Use curves.

Curves are also a modeler's best friend. For the wall seen above, the contours can also be modeled with Bezier curves. Just add a Bezier circle, make the sides straight (V key), and then move the vertices around to the proper places. Add new curves inside the same object for the holes you need. You will see that Blender automatically fills in the shape, and knows which part is solid, and where the holes are. In the Edit buttons, Curve and Surface panel, the DefResolU button determines how many straight segments will make up one curve segment. The higher the number, the smoother the curve will look when rendered. This is one advantage of using curves: the resolution can be changed. In a mesh model, you are stuck with what you first modeled.

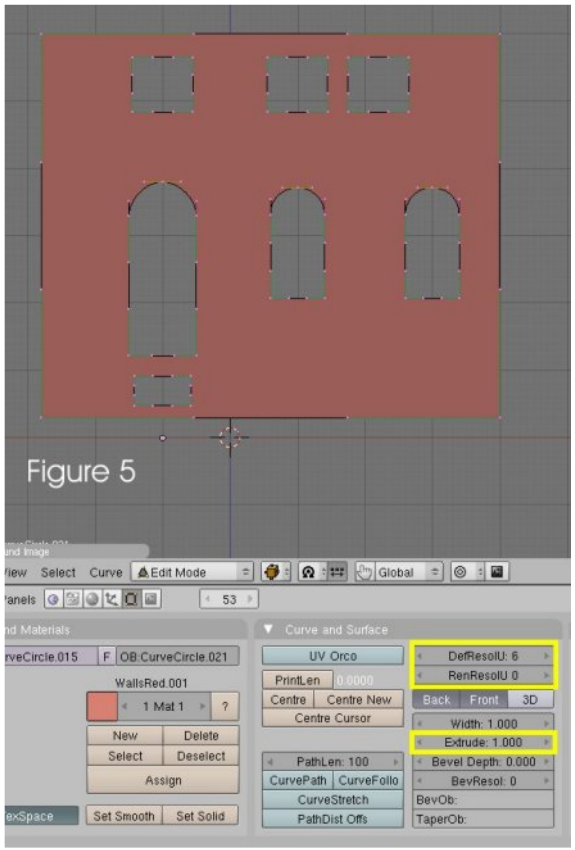


Image5: The Plan made with curves

Don't forget to set a width for the wall. The Extrude button sets the width of the extrusion IN BOTH directions. For example, a value of 1.0 will extrude the curve by 1 BlenderUnit (BU) both ways, which makes a 20 cm thick wall if your scale is 10 BU = 1 meter, the scale I usually use.

Note: Bezier curves are 3rd (some programs also use 5th or 7th) order functions. This means that no matter how many control points you have or how well their handles are placed, a Bezier curve will never be a perfect circle. You'll probably not notice it on a rendering, for archi-viz Bezier is just fine. But if real precision is required, then use NURBS curves, or a mesh with many vertices.

You should also bevel the edges a bit because no wall has a perfect, sharp edge. The Bevel Depth button sets the width of the bevel; this is added on top of the extrusion. So, to have the same 20 cm wall with 0.5 cm bevels you need Extrude = 0.95 and Bevel Depth = 0.05. Set the BevResolution to 2 or 3, that is usually enough.

Beveled curves: Curves can also be used for other architectural details, like railings, rails, gutters, fillets, trims, window and door frames, pipes, wires, ropes, and anything else that is long and has a constant cross-section. This needs two curves: one that gives the length or path of the object (aka directrix, rail curve, path curve, etc.) the other is the cross-section (aka generating curve). The cross-section has to be added to the first curve as a BevOb (bevel object). Figure 6 shows some of the objects that were modeled with beveled curves in this Venetian scene.

The curve at the lower left is in Edit mode, you can see it is a simple square. Its complex cross section is the Bezier curve object called Trim1.

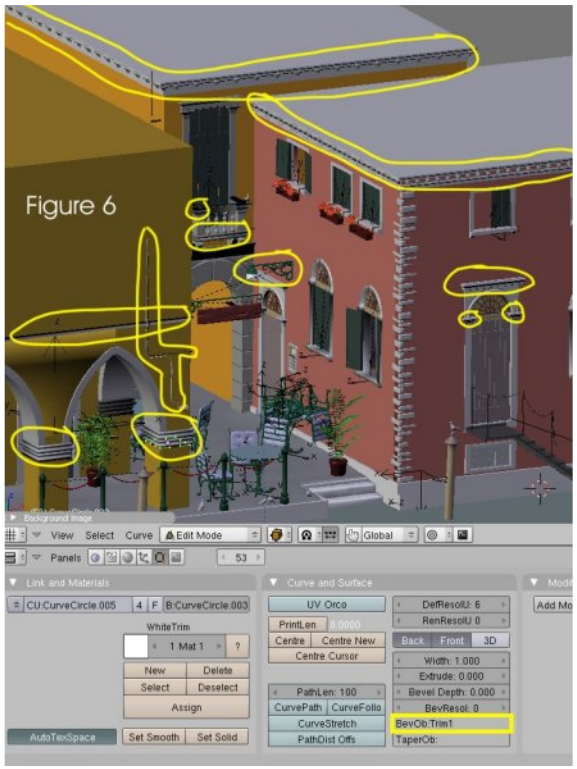


Image6: Beveled surfaces in the scene

Texturing

Program tip: ACME Online
This is a very useful little program. It can be used for one thing only, but at that, it is the best: making brick textures. Instead of the usual "perfect" brick textures, which will scream CG!!! at you from an architectural rendering, this program can make large, non-repeating brick wall textures, with the bricks having slightly different shapes, the mortar oozing out from them and a bit uneven...very realistic.

Note1: Watch the DefResolution of all the curves you use in the scene. The more complex trims in this picture have as many as 20 Bezier curve segments, which multiplied with the default resolution value of 6 is too high, and can boost your render times through the roof. It will not make a difference in quality if you reduce this number to 3 for smaller objects. On the other hand, the path curves might need a higher value if it has large smooth curves in it.

Note2: When the model is done, it can be converted to a mesh (Alt-C) for further detail work, like bevels, some random noise, etc. Remember this is a one way conversion! Also: this might also be needed for other outside renderers if the curve objects aren't exported well with the export script(s).

You can choose from about one hundred brick types, all with different colors and surface textures, plus many mortar colors. Coursing types (how the bricks are stacked): stack, running or third. When you have selected everything, it will render the texture for you. The size can be defined by the user, some very high-res textures I use are 60 bricks wide by 20 high (resolution of 15488 * 1680 pixels). This will cover large walls without the viewer seeing any repetition in the texture, since the program places slightly different bricks - of the given type - randomly.

This is not all though, as you can replace a whole row or just one brick with another type of brick. This allows you to "draw" or "paint" complex bricks setups, including decorative tiles, adding cornerstones and header stones, etc.

The program is free to use. When you want to build a wall with a brick type you haven't used before, it needs to download new data files, so it requires an internet connection. To download, go to http://www.brick.com/, and click on Acme Masonry Design Tool on the right hand side.

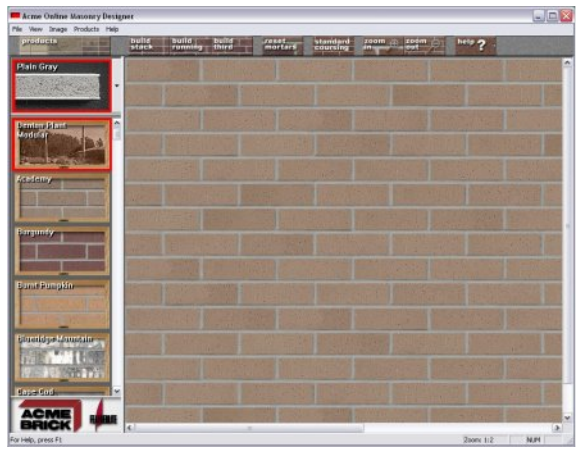


Image7: Acme Masonry designer

Quick Texturing a Large Number of Objects

The scenario is the following: you have a large number of wooden planks (concrete columns, etc.), which you want to texture quickly. You don't want the texture to repeat on the planks. A Python script bundled with Blender will come in very handy: "Archimap UV Projection Unwrapper". First we will need ONE texture. Try to go as high-res as possible. Note that the texture should contain only wood material, if it is of more than one plank, it shouldn't have gaps between the planks, as that would create unnecessary problems using this script. Here, I use a close-up of one wooden beam. This will actually be used for many planks. How? Notice that the texture has many fine, detailed lines; these will be enough for at least ten different planks.

Image8: Plank texture

Getting the model ready: Model the planks, and make sure they are one mesh object. If not, select them all, and Alt-J to join them. Press F or use the menu to change to UV Face Select mode. Select about ten planks. In another window, open the UV/Image editor. In the Image menu, load the texture. In the UVs menu select Archimap UV Projection Unwrapper. See Image9.

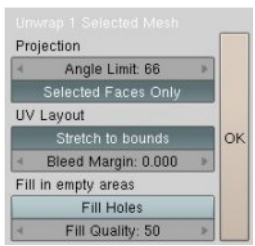


Image9b: ACME pop up

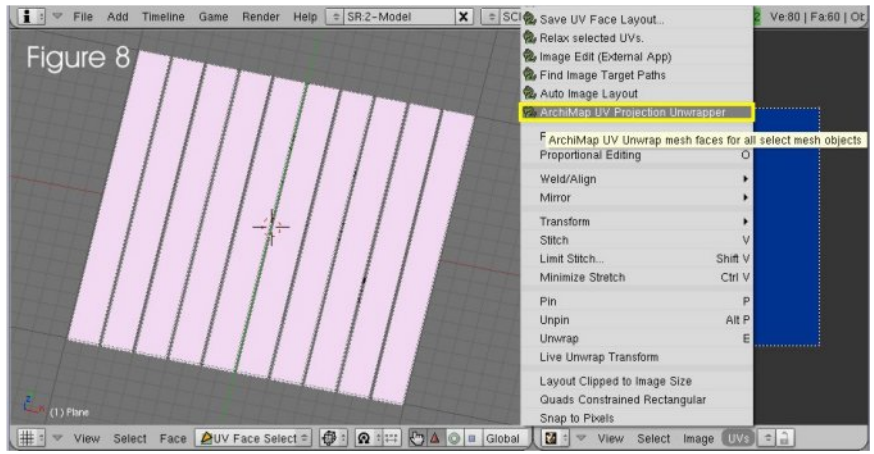


Image9: Acme Projection Unwrapper script

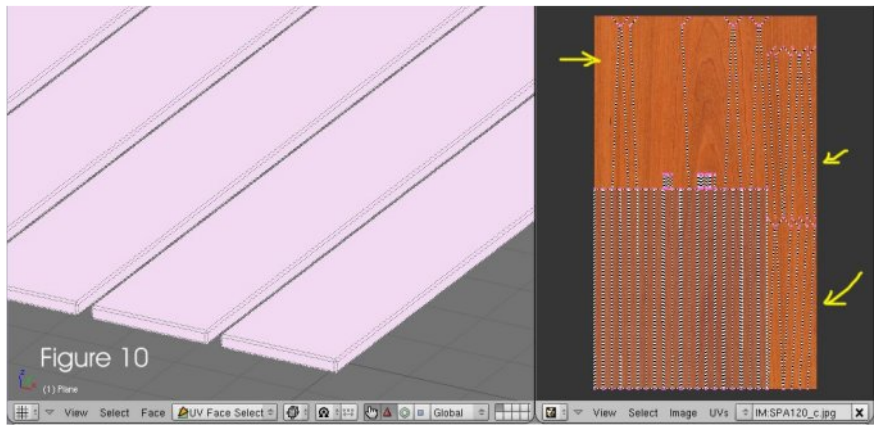


Image10: Default UV generated

The default settings are mostly good. One of the settings that concern us here is the Projection Angle Limit. This specifies the angle above which two neighboring faces will be broken up into separate pieces on the UV map. If the planks are simple 6 sided rectangular prisms (i.e. stretched cubes), then the default 66 degrees is enough to break up the 90 degree angles between faces. Let's say however that you have a single bevel on the model. (This is generally good modeling practice.) Here the faces are at 45 degree angles to each other.

Using the default setting will produce some weird, unwanted shapes (Figure 10, yellow arrows). This is bad for several reasons, for example these thin faces will not be parallel to the wood grain, but some will run across them. But the real problem is that these shapes will lead to a non-ideal placement on the UV texture map. Instead set the angle limit to less than 45 degrees, let's say 30. See the new placement on Figure 11, the left hand side.

The new placement is better, the faces are all parallel to the wood grain, and the UV "islands" are closely placed. However the top half of the image isn't used. Selecting a bit less planks, in my case 9 instead of 10, will lead to a better placement that uses the whole image - meaning that the faces are assigned a larger part of the image giving you better texture quality. You might need a few tries to get the best placement.

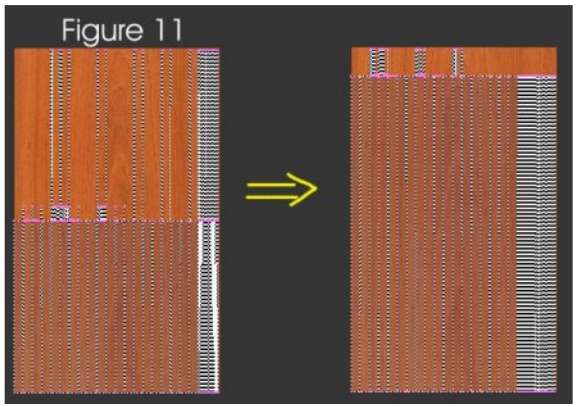
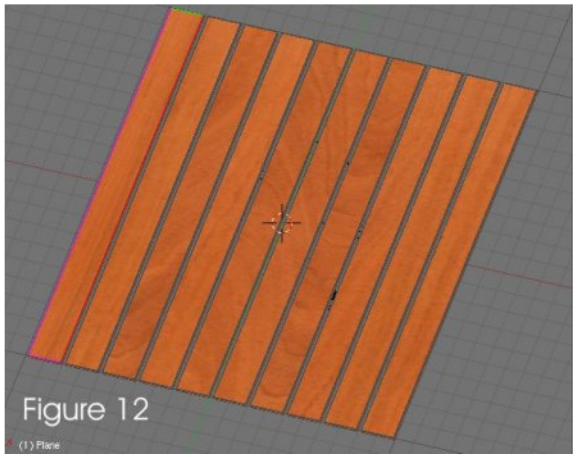


Image9: Adding textres quickely

Quick tip: to select one plank in UV Face Select mode quickly, hover over it and press L. Shift-L will add to the selection.

And here we are! A number of planks textured with random wood textures in, with some practice, less than a minute! ■



NOTE: all of this could also be done by hand, e.g. the UV map on the left of Image 11 could be manually stretched to cover the whole image. The reason for all this experimentation on the number of planks to texture is to automate the process with the above mentioned script. If you have a lot of objects, this type of quick texturing can be done in a few seconds, and will spare you potentially hours of manual UV map creation.

FROM 2D CAD DRAWING TO BLENDER

-by Yellow

Introduction

Architectural CAD files are often criticized for being rather large, messy and containing a lot of duplicate or unnecessary information. Unless you have the privilege of working for a more enlightened practice, with a well-enforced CAD policy, it will most likely be necessary to do an amount of cleanup and preparation, in your chosen CAD application, before attempting to import the files into Blender.

The way you prepare the files may well depend on the approach you intend taking to modeling your building. The choice of how you model your building will probably be dependent on the building's form and construction. As you go through the cleanup process, you'll get a feel for the way the building is put together and a feel for the best way to model it. This can be important if you're working with someone else's design in order to interpret their design correctly. It also helps to get some sort of modeling game plan together, before you go any further.

Whichever way you approach modeling your building, the preparation of CAD files may well include:

* Removing any drawn info which is unnecessary such as text, landscaping, people silhouettes, trees, bushes, hatching and dimensions (specially drawn content, which are made up of hundreds of tiny bitty lines). All the drawn entities in your CAD files, which are imported into Blender, will become editable meshes, made up of edges and vertices. There's no point importing masses of content you don't need, it will only get in the way and slow you down.

* Do a global flattening of z-coordinates, it isn't unknown to have 2D drawings containing stray content with a z-value, the 3rd dimension.

* Scale your prepared drawings to 1 CAD Unit = 1 meter (scaling down by .001, converts millimeters to meters). One of Blender's peculiarities is the restriction it places on your work space. If, for example, you import a large building or site where one CAD Unit equals one mm, Blender will have big difficulties displaying all of it. In fact, you probably won't be able to see much of it at all. Clipping will be evident; panning and zooming will be near impossible. You could try selecting all and scaling down in Blender, if you forget to do it in CAD.

If you're not the creator of the CAD files, you could find that they can be rather inaccurately setup and badly layered. In this case, it can be simpler to draw over the major areas of the drawing you wish to import with your own geometry, divide it up to suit the way you intend modeling, and save it into your own clean CAD file, ready for import into Blender. You could use an unedited import in another .blend file to pick out the finer details later, and Append it into your main model's .blend file, after you've got the basic structure sorted.

The more accurate the CAD drawings, the easier the modeling process will be. Accurate not only in lines meeting each other, but also the definition of areas and perimeters for the different construction materials. Clearly define the boundaries to ensure a smooth process of creating quad faces rather than excessive triangles. In order to use the "mesh2curve" script described later, you'll need to ensure that all boundary lines meet each other (i.e., endpoint to endpoint). The script will fail if there is a break in a boundary line.

Blender's tools for manipulating the edges and verts of your imports are sparse. There is only rudimentary snapping, a proper dynamic snapping feature is sorely missing from Blender, not only for accuracy but for speed as well. So it pays to get the drawings right in CAD with all the tools available to you, then try to work with or correct inaccuracies in Blender later. For small buildings it's common to have plans, sections and elevations in one drawing. If so, set them out in a logical considered way, such as elevations aligned with each other or aligned to the floor plans. After you've removed the unnecessary entities, explode the drawing a few times, to remove any remaining blocks, Blender won't import them unless they are individual lines.

For bigger projects, it may well pay to split your drawings down in a way that suits your modeling gameplan. This may mean creating a series of drawings for floor plans, elevations, and sections. If you intend using the elevations to build your model then it may help to split the elevations down further, with each drawing containing one of the elevational elements. For example, all the masonry walls in one drawing, the windows in another, then create the DXF files for each.

Once you feel you have your CAD files as you want them, save/export as DXF Release 12. At this point Blender's DXF import hack is almost certainly going to fail to import the DXF's correctly.

There are various successful methods to get CAD files into Blender, but all depend on external help and that help is at a cost. You can purchase a shareware file converter application such as 'Accutrans'. Or, there are free CAD applications such as 'ProgeCAD' (export as SVG, but it only runs on MS Windows). There's also Google SketchUP (export as KMZ) but, SketchUp isn't free. You could try the Free Google SketchUP (export as KMZ) but, it's only free for personal use, not commercial.

So, along with decent snapping and measurement tools, a good implementation of a DXF import would benefit Blender's take up for architectural visualization. It is worth the effort implementing, because most CAD applications, in architectural disciplines, use DXF as the common denominator for interoperability between different CAD applications.

The solution I've found to be 100% successful and quick, is to use 'Accutrans', a shareware file converter that can convert many file types, but it's DXF export creates a file that Blender imports without issue, every time. And, the DXF's can be as complicated/detailed as you like. Yes, it's not free or GPL but, it's cheap and does the job perfectly. It even runs on Linux with a little bit of help from Wine.

It is also possible, I believe, to import DXF into Free SketchUP, save as KMZ and import into Blender via JMS's KMZ/KML Import Script. But, don't use the script that comes with Blender, download the updated version from jms's site, version 0.1.9c or above. Version 0.1.9c was the first version to acknowledge that longitude and latitude settings in the KMZ file affected the import scale into Blender so, although the z scale would be correct, x and y scaling could be distorted. If your importing a KMZ that just contains 2D information then, you will need to choose to 'force edges' in the scripts import process.

Using CAD files to create the basis of your model can mean that it is more a process of doing vertex-modeling (i.e. vertex to vertex, creating edges and faces) than subdivision-modeling. Blender's toolset seems aimed squarely at subdivision-modeling at the expense of even simple CAD type tools such as 'proper' snapping, simple measurement, etc. This isn't a suggestion that Blender should be able to do CAD, it's suggesting that certain CAD features are just as valid in a 3D modeler as they are in a CAD application. Once the DXF file is imported, the first thing to do is save it as a .blend file, remove doubles and save again. You're now ready to start modeling.

There are a few implications to importing DXF's to be aware of:

Blender is ok with the first DXF import, however if you then join any of the resultant meshes together, when you try importing another DXF into the same .blend file it is very likely you will get the 'eekadoodle' error or the more professional description that has replaced it. Any further DXF imports will be rejected and fail. However, if you import DXF's one after another, before joining any of the meshes together, you will be able to import multiple times.

This rather defeats the object of splitting the CAD file in the first place if you want to build your model in stages. A solution is to import each DXF into a new Blender session, then create the objects from the import and suitably name them, then save as a .blend file and Append the objects created into your main model as required later.

Another peculiarity you may see as a CAD user is that, when the DXF is imported, the resultant edges, regardless of their orientation, will all have a rotation of 0 degrees. This can be problematic when you want to align a new mesh you create in Blender against an edge from your DXF import when you don't know the angle of edge in Blender. In CAD applications, importing a DXF will put each entity into the 3D world coordinate system and give it a rotation value that you can list or query.
There are various alignment scripts in Blender but, I have found them overly complicated and protracted for what is a simple requirement as a CAD-trained individual, I just want to do a rotate by reference. 3R, a blenderer, has written a python script which allows you to do just that.

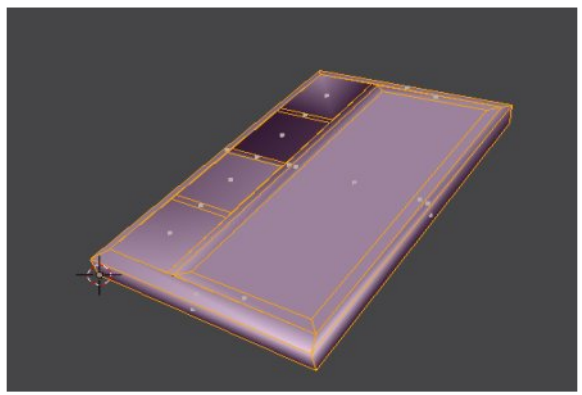


Image1: Non aligned normals

I've noticed a definite difference in the way Blender handles edges between version 2.37a and any version thereafter. Here is an example of a window in Image1.

The difference is that in Blender 2.37a, you could make a face from the DXF import vertices or edges and extrude without getting problems with normals.

In any version after 2.37a, making a face and then extruding results in some nasty normals as seen above. The supposed solution is to recalculate normals, but I've not found this to solve the problem, it just shifts the problem about.

One solution is to extrude the edges first and then make the face, for whatever reason this resolves the normals problem.

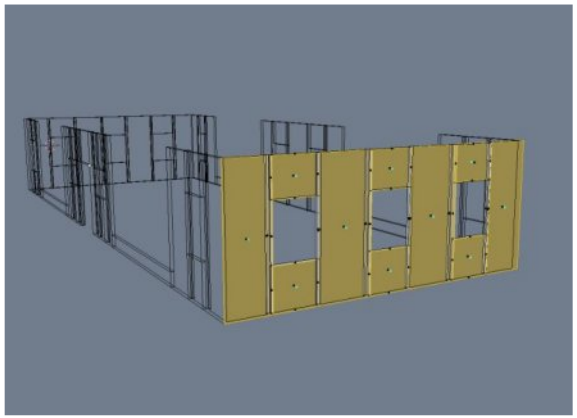


Image2: The extruded wall panels

Unfortunately for me, I find making the faces first and then extruding to be the preferred method and continue to use 2.37a to model buildings before using the current Blender release to do the rest.

You may approach modeling by using the floor plans and 'extruding' the walls up from the plan to the various heights and then, fill in with wall panels above and below door and window openings.

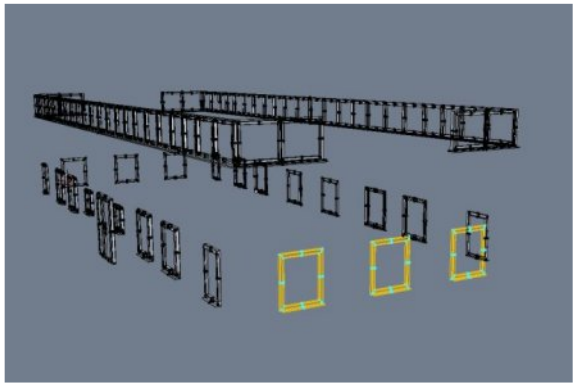


Image3: The extruded windows

Doors and windows could then be added, modeled by either subdivision or working up CAD elevation imports of doors and windows.

You may prefer to use elevation drawings instead of plans and create faces by using the vertices and edges of the DXF import(s) in a sequence such as the following, simplified for demonstration. See Image5.

Image4: The Final Image composed on site photo

First import a prepared elevation just containing brick walls:

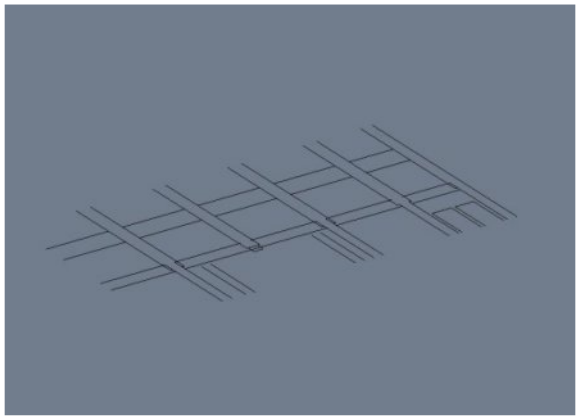


Image5: The elevation for walls

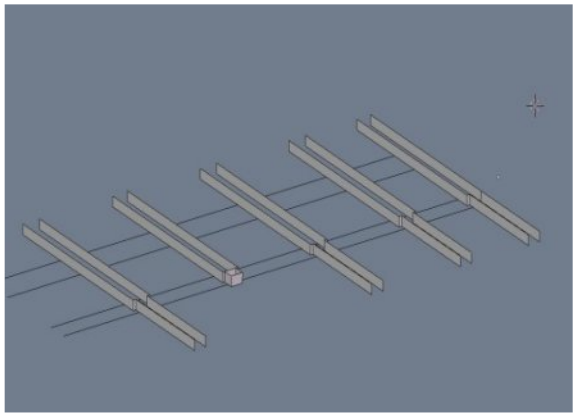


Image6: The extruded columns

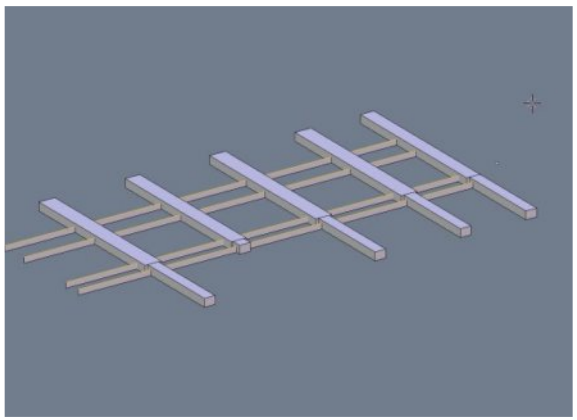


Image7: The extruded wall panel

Continue until wall modeling is completed. See Image8.

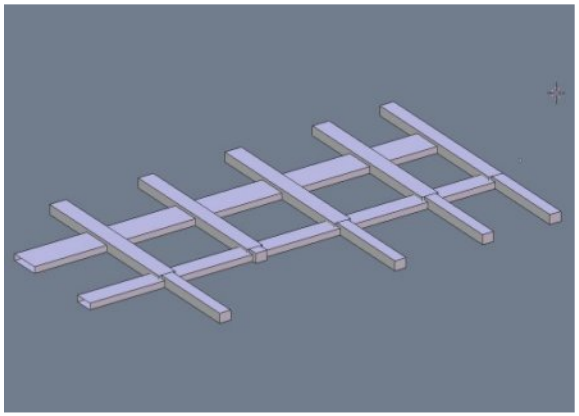


Image8: The Finished wall model

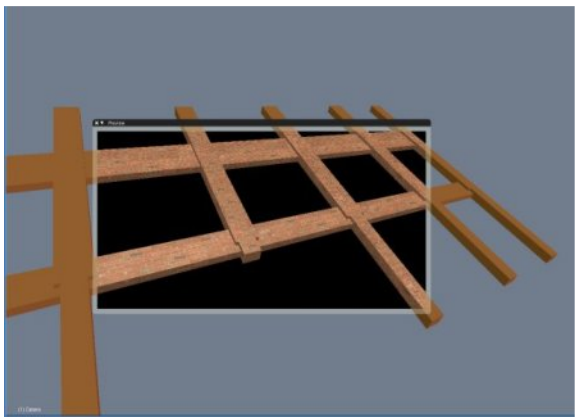


Image9: Wooden texture preview

Apply and scale texture map, preview, adjust scaling and position of map until it looks right.

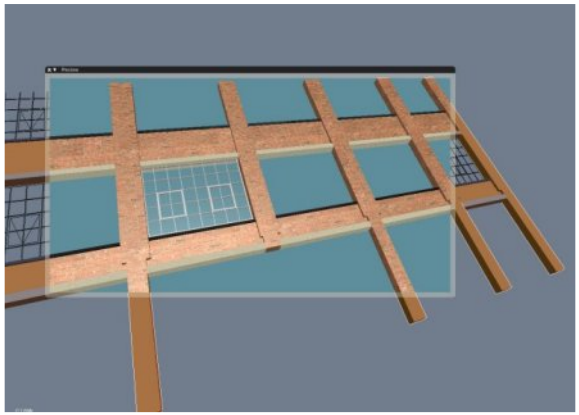


Image10: Corrected texture position

Try to correct brick coursing around door and window openings, especially. Repeat the process for the other elements of the elevations such as lintels, window sills, windows etc. See image10.

An alternative approach to modeling, which can be a fast way to get CAD building elevations with more intricate window openings into a 3D form is as follows:

Image11: The building to model

First import the prepared DXF's. Notice window perimeters are to the outside of the arched brickwork above the window openings and to the bottom edge of the window sill.

The intention is to model the arched brickwork heads and sills separately and in order for the mesh2curve.py python script to work, the outlines need to be simplified continuous shapes. The script is available from 3R's website. See image12.

Run the mesh2curve.py script and watch the meshes convert to curves. Before continuing save

Image12: The wall panels

your work. See Image 13.

Image13: The panels after the script run

Select and join with Ctrl-J and the final wall panels are ready for extrusion via the Curves menu to give a wall with thickness.

The modeling can then continue using the vertices and edges to create faces. Extruding first, if you're not using 2.37a. See image14.

Door and window elevations could then be appended from blend files and added to the

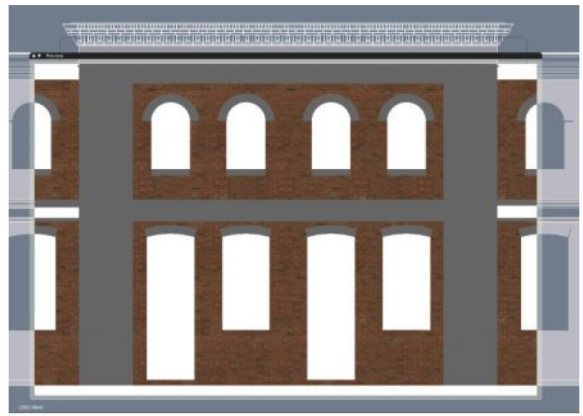


Image14: Wall panels with brick

openings. Other architectural features can then be worked up to complete the elevation. Possibly a roof/eaves/gutter profile imported from the CAD files and extruded around the building to give a roof perimeter line that is ready to start building the roof pitches, etc. The mesh2curve script can be used again for converting meshes to curves for extruding along a path.

Another example, a simple house elevation. For demonstration, I've applied basic textures and lighting to show what can be accomplished very quickly in this way. A quick Yafray render with HDRi lighting.

Image15: The quick render

Import the prepared CAD elevations, including the simplified wall perimeters (1) for running the mesh2curve script on.
Model the doors, windows, etc. using the edges and vertices from the full-detail CAD elevations (2) and combine the extruded walls created from the mesh2curve script with the individual door and window objects created from the full-detail elevations(3).

Here are some images of another house model, done in the same way. Images are rendered with Yafray, skydome, low, cached and HDRi image

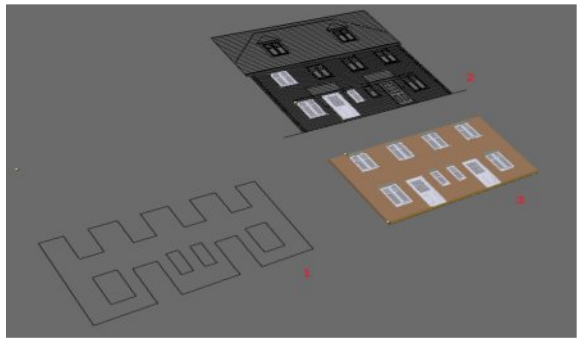


Image16: Elevated doors and windows

for lighting. They are unfinished works-in-progress as you can see. I am working on creating a library blend file of trees, bushes, planting, etc.; textured and ready to Append into my scenes.

Image17: The rendering of the house model

Plant Studio is now free and there are many plant and tree parameter files for download. The trees in this image are Aspens made with Arbaro using the default settings.

All the elevational elements to this house, for example the dormer windows, canopies, roof lights, entrance doors, window styles and standard window sizes, are all objects stored in the library blend files I've created.

So, I can quickly build the walls to a particular house as described above and then append in the objects I need, from the library files (including a texture library), to finish the elevations.

Image18: Bird view render

This process works well for visualizing house builders' house types that are all variations on a theme, using a set of standard details.

When appending objects created previously, it can be useful to use a duplicate of your DXF import building elevations as a template to snap your appended objects to in order to accurately locate them. In the example below, I used the template to locate the canopy and it's brackets over the entrance doors and the dormer window positions in the roof.

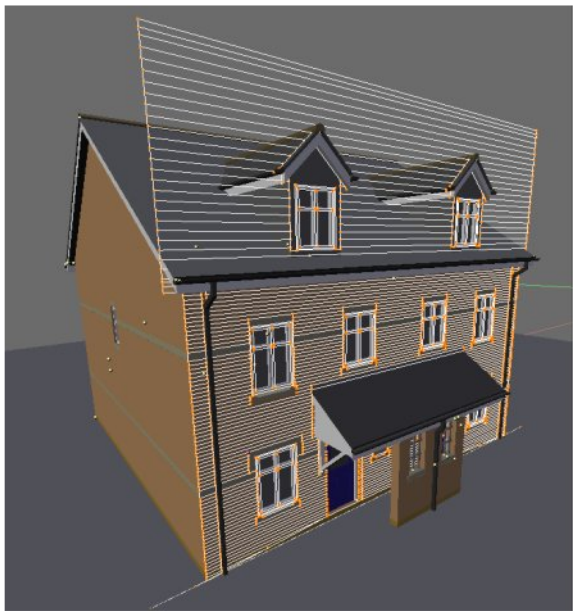


Image19: The house template

You can model the basis of a landscape for the site, on which you intend putting the buildings, in the same way. Import a cleaned up 2D DXF of the site survey, having first defined the perimeters in the CAD file for the various landscape elements like the site boundary, the area to be planted and turfed, the outline of the roads and footpaths.
Then use the mesh2curve script on each of those perimeter lines to convert them to closed curves, giving you solid infill to your defined areas.
Mesh (1) below is the perimeter for the landscaping. Mesh (2) was a full site import, but I have been moving the meshes edges onto the final site model and then modeling into 3D. The final site model was based on the curves generated from mesh (1) by the mesh2curve script.

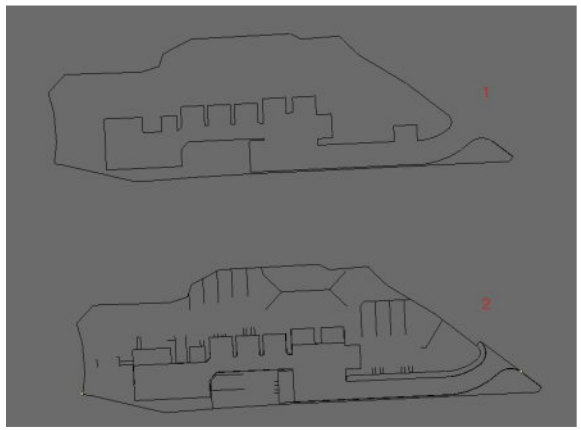


Image20: A Siteplan

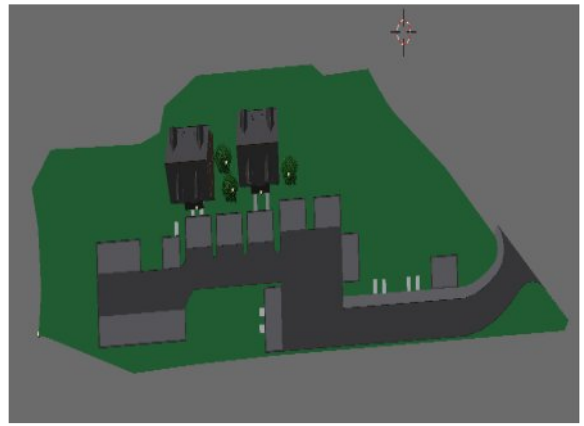


Image21: Site plan after solidifying

As an alternative to modeling the landscape of the site, where perhaps you're just doing a massing model and want to show the building in relation to it's surroundings, you could just use an image file as your site, perhaps colored up in The Gimp or Photoshop. Here's a method:

In your CAD application, open up or create the proposed site plan. For example, from an ordinance survey, clean it up as described previously and then add a square border to form a frame around the area of the site you wish to show in your site image map.

Save the CAD file as a DXF and put it through the conversion process described earlier.

Then in CAD, print the CAD drawing to an image file. I use .PNG and use the border as your paper edge. In AutoCAD, I've setup a 5000x5000 paper size and then I select by window the square border. I also use quite heavy line weights to help the final image show up clearly when used in Blender. The resultant image could be colored up in The Gimp to form an attractive site plan.

In Blender, import the DXF, remove doubles etc. Use two opposite edges that form the square and create a face. Then create a material with an image texture for the new face, the image being your site plan .PNG. Set the image to Clip rather than Repeat. The image will hopefully now be registered accurately over the imported site plan mesh. You can now extrude or model some of the surrounding buildings using the building footprints of the site plan mesh. Add your materials, lighting and camera. When rendered, the modeled buildings should appear to sit accurately on the site plan image map.

(1) DXF Import of site plan.
(2) Plane with image map of site.
(3) SketchUP model.

And a quick internal render. Work in progress again.

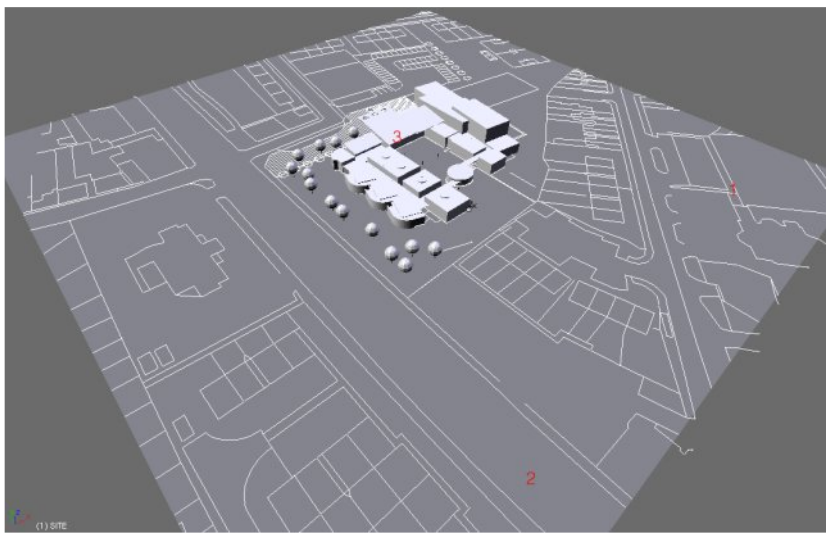


Image22: The Siteplan progress

Looking at an architect's office situation, where it's not unusual to have an architect or building designer conversant with SketchUP, dealing with design development and design issues, whilst a 3D artist or CAD technician conversant with AutoCAD and Blender, could be modeling and visualizing the site from DXF imports ready for integration of the finalized SketchUP model when its design is finalized.

The SketchUP model could be imported as a KMZ, as described earlier, and accurately located on the site by snapping it to the previously imported site mesh.

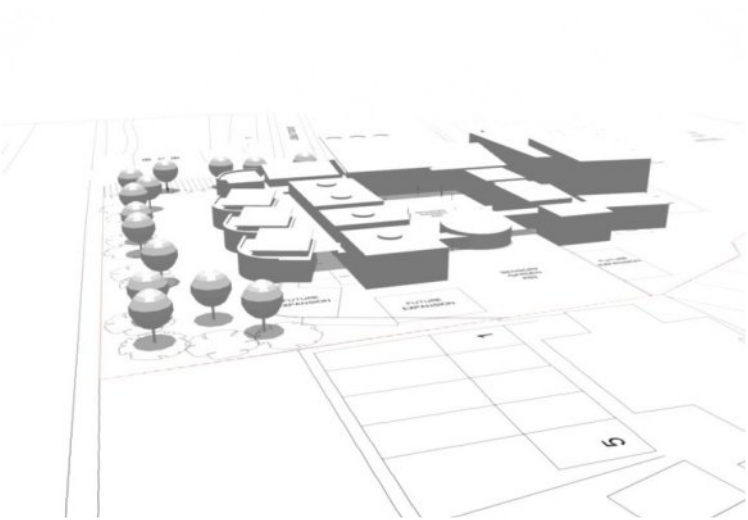


Image23: Siteplan quick render

Conclusion

My intention for this article has been to suggest ways Blender could be introduced into an architect's work flow, working in conjunction with CAD, the major software element of any modern architects office. There are many other ways Blender could, or in some cases, is being used by architects and, as the subject is so wide, this article only scratches the surface of the opportunities available. ■

The Making Of The Cathedral

- by SebestianKönig (stulliDPB)

Introduction

I was asked to write something about my project. However, still being rather new to Blender, I can not write a "How to Build a Church" tutorial. Instead of that, I'll write about one year of my working with Blender.

So, here's the story of my cathedral.

It started all back in July 2005. I was still a student of Education of Art at the Academy for Arts and Design in Halle / Germany, and had to do my university degree. I was sick of making art and wanted to do something special. My teacher told me about an exhibition about Cardinal "Albrecht von Brandenburg", which should be opening in September 2006. He said the museum might be interested in some kind of software for a computer terminal that would stand inside the museum and I thought: "Well, this might be more interesting than doing just arts, drawings and paintings".

I had barely messed around with Flash, Photoshop, and – hold on to your seat – PowerPoint (which still is some kind of magic program to some of the students and teachers...). But, I was a bloody noob to Blender!

So, I started from scratch. I chose "Director MX" to do the programming. Believe me, I still regret this. But lucky me, that I chose Blender to do the cathedral. In July, when I began to look for the proper software, I tried Maya, Studio Max and Cinema 4d. But somehow I got drawn towards Blender and I did not regret this choice for one single minute.

The project's main aspect was to reconstruct the cathedral of Halle (Germany) as it looked in 1525. Today, it is a rather boring church, with some statues, one altar and uncolored windows. But, in the beginning of the 16th century, the church was filled with hundreds of paintings, tapestries and golden relics.

In those days, Cardinal Albrecht von Brandenburg was about to inaugurate the cathedral and a cycle of 18 altars, mainly painted by Lucas Cranach the Elder. Most of those paintings are lost and gone today, many of them scattered to the four winds. The main purpose of my work is to show where those altars stood, what the jube might have looked like and what the ambiance inside the old church might have been.

There were several problems to deal with. The first one was me. I knew how to make a cube, how to extrude, how to move vertices and faces, and how to subdivide. And, that's what I did. Soon the rendertimes where increasing rapidly because of hundreds and thousands of double-vertices, double-faces, dozens of doubled and tripled procedural textures, high-resolution texture-images...well, you know what I mean.

The second problem was that I had no plans or blueprints at hand, only two old drawings of the church, a floor plan and a sheer plan. Both were hand drawn and very inaccurate.

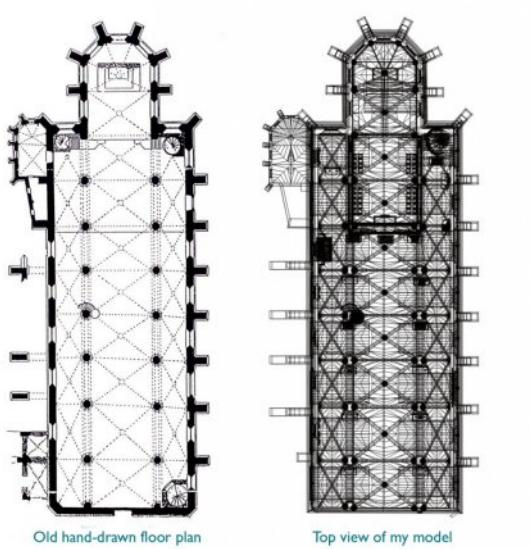


Image1: Left Hand drawn model, right top-view of the finished model

But, never the less, I advanced quickly. Soon I had the walls, the windows, the floor and the roof, so I then put some textures on them. I admit, I was very proud of myself. But as I mentioned above, the models were very unclean and most of the textures were useless.

The third and main problem was that there were no information or pictures of the cathedral's appearance in 1526. The jube doesn't exist anymore, nor does the west-gallery with the renaissance organ in it. The windows are plain white today, but back then, they were made of stained glass. Only a few old documents describe roughly where the altars were placed. In co-

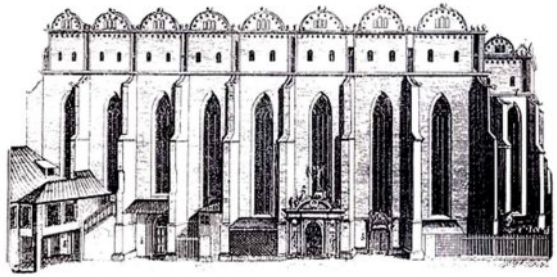


Image2: Old side view plan

operation with the museums art-historian, I rebuilt the cathedral and placed the altars.

The most difficult part was the lightning. I had to find the balance between "looks good" and "I can see everything". I spent hours and hours rendering and tweaking light. I tried HDR and Ambient Occlusion, multiple area lamps with soft-shadows, multiple volumetric spotlights with light-textures and so on.
…
Either it looked good to the right and bad to the left, or the render times were to long, or it was boring.

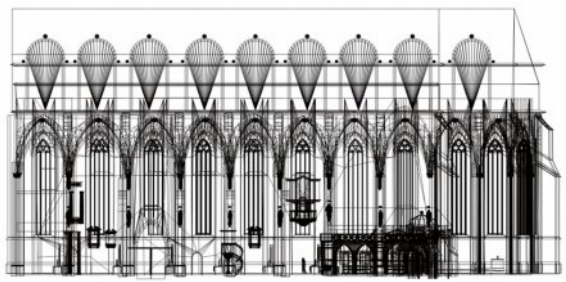


Image3: Finished side view

Eventually, I found the perfect balance between awesomeness and reasonable render-times. It was the UV-Mapping that solved most of the problems. But, before I could start to UV-map, I had to re-model everything to have clean meshes that allowed a correct placement of the textures. I started to remove doubles, make clean faces and proper vertices (I don't know how often I pressed Alt+M, but believe me, it was very often…).

And again, I was very proud of me. But still it didn't look so good.

Two months before the deadline, I discovered how to UV-Bump Map and, suddenly the church began to look REAL! All those boring walls looked good at once and the lighting was much easier. It turns out that I did not need any shadow-casting lamps, no soft-shadows, just Ambient Occlusion with SUB enabled.

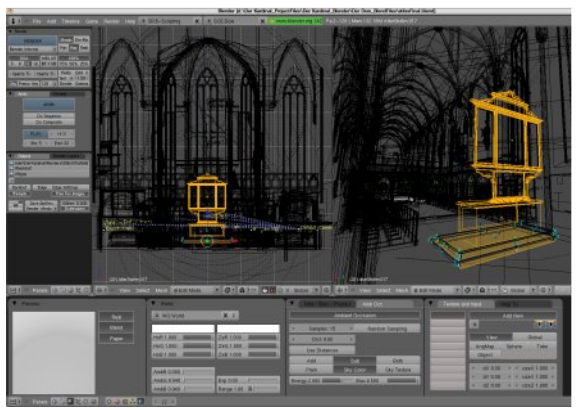


Image4: Work in progress

Image5: Prelen Altar

Now that the cathedral is finished, almost everything is UV-mapped.

If I had taken a closer look at the UV-option earlier, it would have saved me a lot of time. But like so many others, I was somewhat afraid of UV, because it looked so complicated. It's not. It's easy. I really would like to encourage everybody to try it out.

It was a really good year working with Blender. It was never annoying or boring. Blender runs smooth and comfortable, the workflow is great - but then again, I have nothing really to compare it with, like AutoCAD or StudioMax. I'm not an architect. No one could build that church after my model. The model is not accurate, but I think that everyone can imagine now what the cathedral might have looked like when Albrecht von Brandenburg inaugurated his church and the cycle of altars. For that purpose of visualization, Blender was perfect.

Now I'm addicted to Blender. Director MX is still driving me mad. All in all, I'm an artist and not a programmer. Python is still a mystery to me, but I bow to all those scripting and coding heroes who make Blender possible.

After one year of working on the cathedral, there are still things I could improve, but all in all, I'm glad I did it. I think it worked out well. One of the reasons I started this project was the thought of "I can do that, too. BLENDER can do that, too." I hope I was able to prove both. ■

Image6: Engels Altar

Regards and Happy Blending!
Sebastian König // stulliDPB

On (08.09.2006), the exhibition was opened. I managed to finish the program with DirectorMX, insert all the animations, Quicktime-panos, pictures and texts, and to my surprise it works very well. The terminal is standing inside the church, and many visitors looked at it. Even Cardinal Lehmann, the chairman of the conference of the german bishops, who came to open the exhibition and held a speech, watched the animated walkthrough for several minutes and I think he liked it.

Folks, Blender has reached the Vatican!

Image7: Presentation

The Making Of Burly Brawl Scene

- by Mike Pan

Introduction

The decision to re-create the infamous Burly Brawl arena, from the movie Matrix Reloaded, came when I was tinkering with the idea of exploring and showcasing the possibilities of the Blender 2.42a game engine. But, to call the finished project a game is rather euphemistic. By the end of the project, all I had was an empty arena waiting to be filled with action. Nevertheless, it shows how easy it is to create a very realistic-looking 'game map' that runs at more than 100 fps (frame per second) in the Blender Game Engine.

By watching the DVD fight scene, and looking up images on the net, I gained a pretty good idea of EXACTLY how the arena looks. The modeling was very straightforward, a few simple cubes here and there, combined with some extrusion and resizing, the arena quickly materialized. Since the viewer(camera) will only be limited to the ground of the courtyard, there is no point in modeling the 'other' side of the buildings where it won't ever be seen. This is done in order to save development time and improve the performance of the game. See Image1.

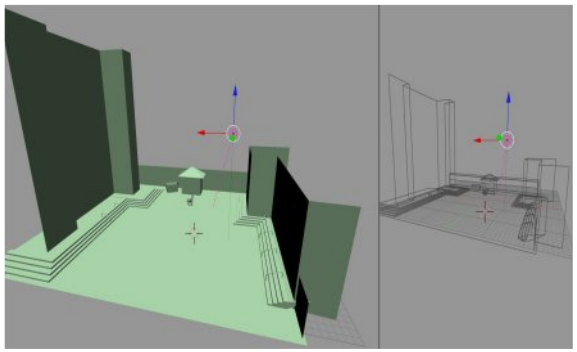


Image1: The scene

The high-rises are created using the new Array Modifier in Blender 2.42 and it allows quick duplication and extrusion required to create the many floors of a building. See Image2.

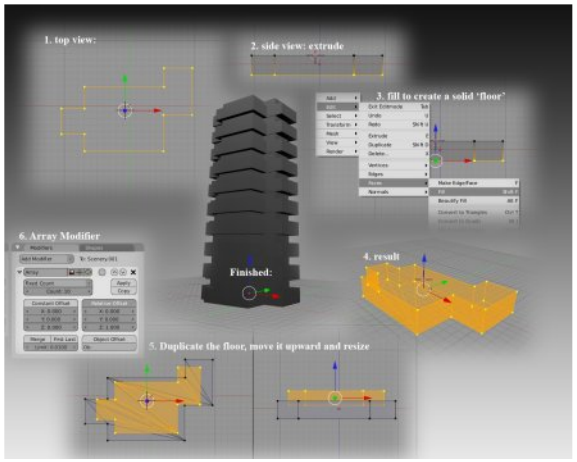


Image2: The High-Rises

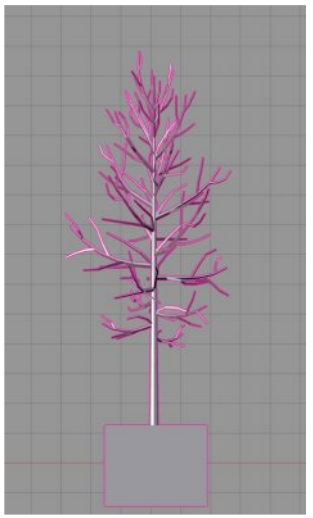


Image3: Gen3 Trees

The trees are generated by the wonderful Gen3 script, I disabled the generation of leaves to give the scene a more deserted feeling.

The finished scene, with no textures attached. Notice the presence of many seemingly orphaned polygons. They are, in fact, building walls waiting to be textured.

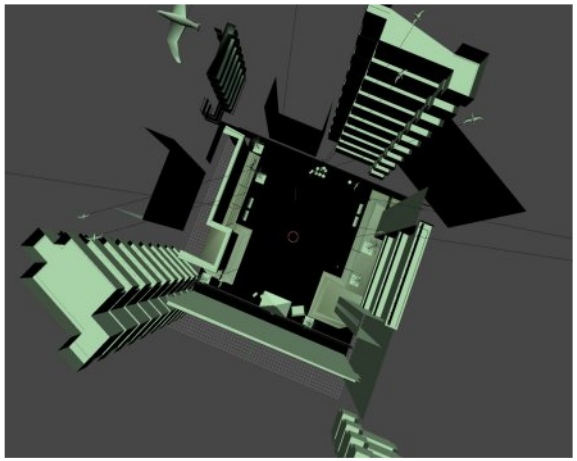


Image4: The finished scene

Texturing is done by enabling the "Show Blender Material" option from the "Game" menu. This allows the artist to use an enhanced version of the real-time material system, featuring multi-texture blending, vertex lighting and even GLSL support. Using the staircase as an example, the first layer is a stone-like image texture, while the second layer is 'overlayed' on top of the first layer of texture. The purpose of the second layer is to create a more varied appearance, breaking up the obvious patterns of repetition of the first layer. See Image5.

The actual building textures came from the free texture site [www.mayang.com/textures]. For the ground of the courtyard, I added a 'shadow' layer which consists of a

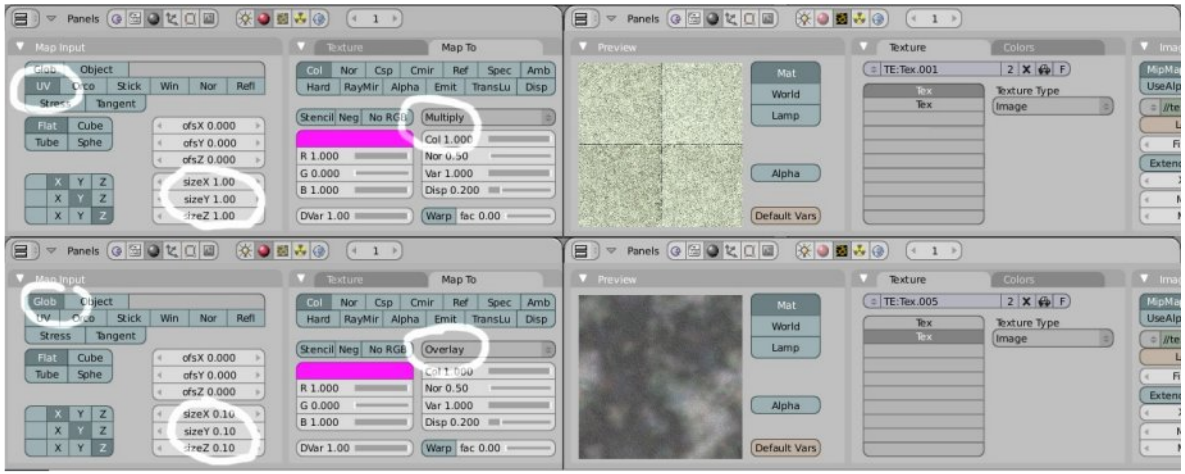


Image5: Texture settings for the scene

pre-rendered ambient-occlusion map. This adds a realistic looking soft shadow to the ground texture, which ties the scene together nicely. See Image6.

More info about the Burly Brawl scene can be accesed from http://mpan3.homeip.net/sub.php?mid=games.

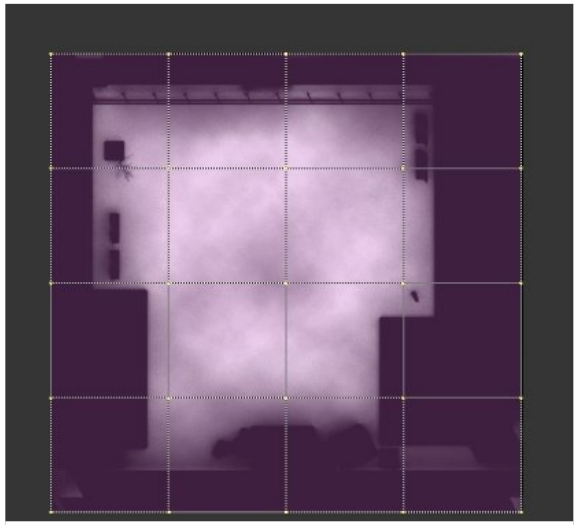
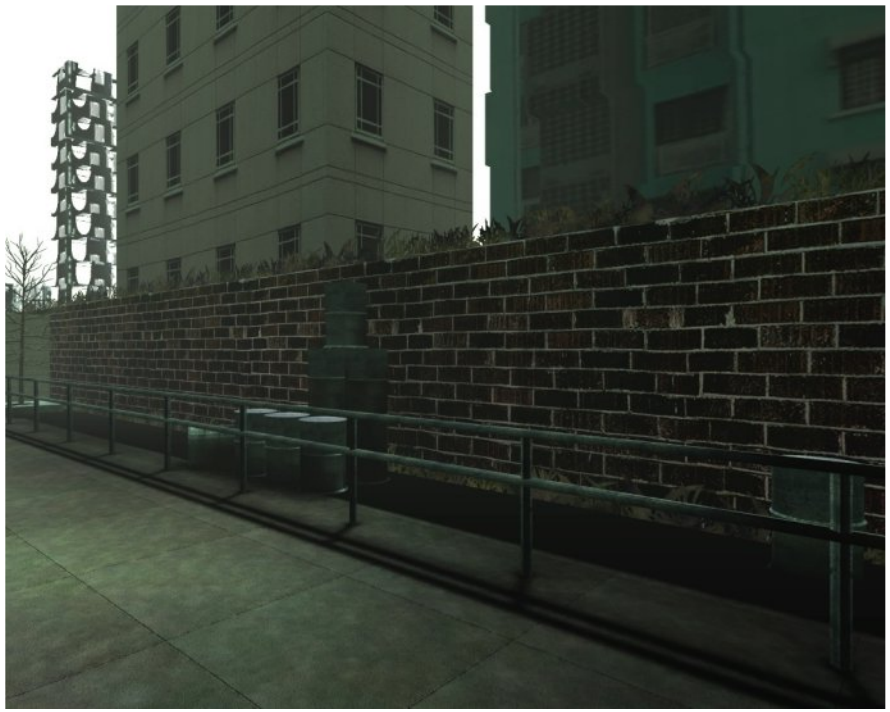


Image6: The floor shadow texture



Mike 'mpan3' Pan http://mpan3.homeip.net

I am 17 years old and have been using Blender for just over 4 years. As a self-taught graphics artist, I do most of my modeling, stills, animations and game creation in Blender. I also have a passion for realtime graphics and photography.

Interested in writing articles for BlenderArt Magazine?

1. **We accept the following:**
   - Tutorials explaining new Blender features, 3d concepts, techniques or articles based on current theme of the magazine
   - Reports on useful Blender events, throughout the world.
   - Cartoons related to blender world.
   - Interviews of well known Blenderheads.

2. Send submissions to *sandra@blenderart.org*. Send us a notification on what you want to write and we can follow up from there.

   Some guidelines you must follow.
   - Images are prefered in PNG but good quality JPG can also do. Images should be seperate from the text document.
   - Make sure that screenshots are clear and readable and the renders should be at least 800px, but not more than 1600px at maximum.
   - Sequential naming of images like image 001.png... etc.
   - Text should be in either ODT, DOC, TXT or HTML.
   - Archive them using 7zip or RAR or less preferably zip.

3. **Please include the following in your email:**
   - Name: This can be your fullname, nickname or a name of your choice.
   - Photograph: As PNG and maximum width of 256Px. (Only if submitting the article first time)
   - About yourself: Max 25 words.
   - Website: (optional)

   Note: *All the approved submissions can be placed in the final issue or subsequent issue if deemed fit. All submissions can be cropped if necessary.*

During the second week of July, there was a summer camp at the North Carolina State University called Redhat High. A pilot program aimed specifically at financially disadvantaged 8th and 9th graders, this camp was created for several reasons. The execs at Redhat appear to be concerned with a sharp drop in technical skill interests among our youth as they leave secondary school and enter high school.

They also want to indoctrinate the students into the Open-Source community, showing them what they can achieve without anything but their will and a little brain work. This pilot program was watched with great interest as they plan to expand it overseas into India, Bangladesh and several other countries worldwide.

Michael Tiemann, CTO of Redhat, (also co-founded Cygnus, President of the OSI, your basic Open-Source god), got the idea to involve Blender in this project. They had four tracks for the students to choose from: Audio, Video, Web Design and 3D Modeling/Animation. The other tracks all used Open-Source software of course, ranging from Cinelara to Audacity to The Gimp, and Blender was the perfect choice for 3D.

Mr. Tiemann posted on the Blenderartists.org forums and three blenderheads responded. Jason VanGumster (Groo/Fweeb) took the helm and Jeffery McGregor (Enzoblue) and Jonathan Williamson (mr_bomb) also signed up as teachers.

Of the four tracks, 3D Modeling/Animation was the only one with outside volunteer support. The other tracks were handled by Redhat interns and employees, as their massive training complex is walking distance from NCSU. Our fearless blenderheads were treated with plane tickets, dorm rooms (though Groo got a hotel because he had a car), and all the food they could eat. Once at the NCSU campus, they were met by Claire Sauls who was hired, nine months previously, to handle all the particulars and to shepherd all fifty-plus kids to and from their classes. She also arranged events for them (bowling, movies, water-balloon fights etc.) and was the busiest woman on campus. Each track had a Redhat helper too, and together they kept things running smoothly and the kids had fun.

The Blender classes were held on the Redhat campus at the engineering building. In a room with thirty desktops running Fedora, these fourteen kids delved into their projects with zeal. Using only a projector hooked up to Jonathan's laptop, Groo, Jeff and Jon walked the students through basic interface and very light modeling tutorials during the three-hour class. The intent was to have them all get something to show their parents at the final presentation that was held the following Saturday morning.

Once walked through the basics, all kids were encouraged to come up with ideas as to what they wanted and fine tune them as much as possible as the week progressed. Hurdles came about with power outages and changed passwords, but the kids learned at an almost frightening rate.

By the end of the week they were working with light animations, particles and texturing, and some even had full armature rigs going. It was clear by Friday that they were ready, and Groo stayed up late rendering their projects while Jonathan fought with his Mac to make up a movie presentation.

The 3D track was also watched closely by the Redhat to see what Blender could do and how easy it was to use. Groo was invited to the local Triangle Linux User's Group, (TriLug), to show off Blender's functionality and gave a 90-minute speech there. At the graduating ceremonies, parents were treated with CD's of all the work the kids did. The CD's had copies of Blender and several tutorials and links - a basic starter package. Our Blenderheads also collaborated on a short animation in their off-time, which is bound to happen when you stick three enthusiasts in a room for a week. They put the animation on the gift cd's too, along with all the blend files of course!

Basically, Blender got off to a good start in North Carolina as Redhat was very impressed. The kids

loved it, and parents were stunned at the things their children were able to make. The parents also gave many heart-felt thank-you's to our three heroes and they left feeling that they definitely made a difference.

So, keep an eye out for RedHat High! There may very well be one coming to your local college sometime soon... ■
http://blenderartists.org/forum/showthread.php?t=7223

- EnzooBlue

Introduction to Character Animation

Author - Ryan Dale

"Introduction to Character Animation" is one of the ten projects chosen for this year's Summer of Documentation project.

Ryan took the approach that the best way to learn character animation was to actually animate a character. In doing so, he created one of the largest and most comprehensive tutorials that I have seen to date. In an effort to make this tutorial accessible to the widest range of users, Ryan introduces new topics and concepts as side notes that beginners can read and advanced users can skip over.

He starts off with step-by-step instructions for making the character you will be using throughout the remainder of the tutorial. Once the character is modeled, he then shows you how to apply materials/textures.

Realizing that not everyone would be interested in modeling the character from scratch, Ryan supplies a blend file download for those who want to jump ahead and get straight into the animating. In the next section, he introduces the various aspects of rigging the character for animation. He covers envelopes vs. vertex groups, the different methods of adding IK constraints, the troublesome area of elbow and knee bones and foot rigs, and then goes on to explain the use of the fairly new "Stride bone".

Once your rig is built, you are shown how to skin/weight paint the rig to work well with your character. There is even a section on custom bone shapes.

Since the goal of this tutorial is to end up with a short animation, and lip-syncing is often required, Ryan shows you how to set up Shape Keys and some basic shapes for the lip-synced portion of the animation. The next segment covers the various areas of lighting and camera setups used for the final animation.

The final segment of the tutorial covers the nuts and bolts methods of animation. To create the finished animation, you will be learning how to use the timeline, IPOs, the action window, setting and adjusting keys, adding sound and how to blend it all together in the NLA editor.

As you start this tutorial, don't expect to work through it in just a few hours. At over 150 pages of printed material covering everything from modeling to final animation, this project could easily take days to even weeks to complete. But believe me, it will be time well spent.

The amount of information presented is amazing and Ryan did a wonderful job presenting it in such a way as to make it easy and painless to learn such a complicated topic.

While I consider myself a better than intermediate user, I learned quite a bit while doing this tutorial. It also brought me up to speed on the newer features that I had not yet explored. ■

-blenderart

Zsolt - Interior

Zsolt - Church

Oscar Alvarado - Terraza

Oscar Alvarado - The Temple Of Kalion

Oscar Alvarado - TV Room

Cristian Mihaescu - Abandoned City

Cristian Mihaescu - Romeo and Juliet

Cristian Mihaescu - Station

Yellow - Hall

Zooly - Stair

Issue 7 November 2006 - Anniversary Issue!

Theme: Blender 2.42 Features Buster!

- *Blender Materials*
- *Blender Noodles*
- *Compositing*

Issue 8 January 2007

Theme: Car Modeling Mega Special!!

- *Modeling Cars*
- *Modeling Tires*
- *Car Paint and more...31*

- Think you have some proficiency in writing articles?
- Want to contribute articles and share your knowledge with blenderheads around the world?

Learn how to contribute to Blenderart Magazine here!!

Disclaimer